Implementation, Validation and Profiling of a Genetic Algorithm for Molecular Conformational Optimization

Victor Gueorguiev
Department of Computer Science,
University of Cape Town,
Private Bag X3, Rondebosch, 7701,
South Africa
GRGVIC001@myuct.ac.za

Michelle Kuttel
Department of Computer Science,
University of Cape Town,
Private Bag X3, Rondebosch, 7701,
South Africa
mkuttel@cs.uct.ac.za

ABSTRACT

Prediction of the lowest energy conformation of a protein chain is a challenging optimization problem in computational chemistry and biology. Simple lattice-based protein models have been shown to be effective representations of the characteristics of proteins important in protein folding. An effective genetic algorithm for conformational optimization of proteins represented by the hydrophobic-hydrophillic lattice model was recently published. In this work, we create a publically available implementation of this genetic optimization algorithm. Tests of our implementation show equivalent performance to that reported for the original, in terms of both optimal conformation and number of function evaluations. In addition, we test our implementation across a range of data set sizes to characterize the performance of the algorithm as chain length increases: benchmarking that is necessary for future optimization and parallelization of the algorithm.

CCS Concepts

• **Computing methodologies→Molecular simulation**

Keywords

Genetic algorithm; HP Lattice; conformational search; energy optimization; hydrophobic-hydrophilic model.

1. INTRODUCTION

Determination of the three dimensional structure of an arbitrary protein from the sequence of its constituent amino acids has been identified as one of the ten most sought after solutions in protein bioinformatics [1]. Most proteins fold rapidly into a well-defined single low-energy conformation under physiological conditions. The conformation of a given protein is of interest because it largely determines the protein's biological function. Thus, knowledge of the three-dimensional structure of a protein can assist in understanding disease mechanisms and hence inform drug design strategies.

SAICSIT '16, September 26 - 28, 2016, Johannesburg, South Africa.

DOI: http://dx.doi.org/10.1145/2987491.2987529

The theoretical prediction of the structure of a specific protein chain requires both an effective model of the protein structure as well as an efficient optimization algorithm. The most accurate atom models are quantum-mechanical. However, although the interactions between atoms are governed by quantum mechanics, currently it is not feasible to perform a precise quantum-mechanical conformational optimization for large molecules such as proteins. Therefore, theoretical models commonly used for conformational modeling instead use classical mechanical approximations. These molecular mechanical models range from complex all-atom representations with a force expression representing interactions between atoms, to coarse-grained (but effective) lattice models where each amino acid monomer is represented as a single unit.

Lattice models simplify the molecular representation by both removing atomic detail and discretizing space as a lattice. The simplest model, the hydrophobic-hydrophilic (HP) lattice model, represents a protein as a series of connected amino acid monomers on a 3D lattice. All constituent amino acids are classified as either hydrophobic (H) or hydrophyllic/polar (P) monomers. Despite their simplicity, lattice-based protein models have been shown to exhibit key features of the protein-folding mechanism [2] and are an effective simplification for studying this process [3].

However, even with the simplest models, the conformational space available to a protein is enormous. As a result, an exact solution to the conformational optimization problem rapidly becomes infeasible as the length of the input protein chain increases: the folding problem for the HP-model has been demonstrated to be NP-complete [4]. More tractable approximate search methods are therefore employed for optimization. Several biologically-inspired algorithms have been applied to the HP-lattice optimization problem, including memetic algorithms [5,6], particle swarm optimization [7], a contact interactions method [8], an immune algorithm [9] and ant colony optimization [10]. Custiodio et al. recently reported a related approach to optimization of HP-lattice models using a genetic algorithm, termed the "adaptive genetic algorithm with phenotypical crowding" or GAHP [5]. This method employs a "crowding" parental replacement strategy which forces competition between the most similar individuals in the population. This approach allows for the formation of niches and thus the preservation of genetic diversity in the population. As the GAHP algorithm showed promising results – increased performance (as measured by fewer function evaluations) compared to competing algorithms and improved solutions (as measured by the number of H-H contacts) – we wished to use it for future work. However, despite

the fact that the original paper states "GAHP's source code will be made available on request to the academic community", repeated requests to the authors for the code were not forthcoming. Therefore, in this work we created an implementation of the GAHP method. Our goal was to create a correct implementation that was freely available to the public and which can be used as a platform with which to test and profile the method for performance improvements, including parallelization for execution on multiple cores. Our implementation is here validated through reproduction of the tests described in the original paper. We then extend the tests of the method with both larger and smaller data sets, and, finally, profile the performance of the method across a range of protein chain lengths.

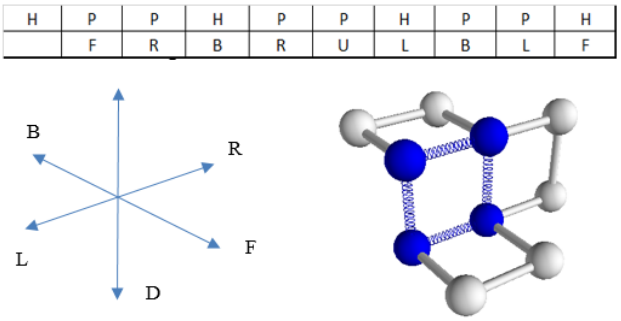

Figure 1:An example of the HP lattice encoding scheme. The blue spheres are H monomers and white are the P monomers. The white connectors indicate the molecular chain and blue connectors indicate the HH contacts. Here the fitness of the molecule is 4 (4 contacts).

2. METHODS

Our implementation of the GAHP algorithm is here termed AGAHP (Alternative implementation of GAHP) to avoid confusion with the original. Our implementation is freely available on Github at https://github.com/BrutishGuy/ADAGP. In our development, we closely followed the methods described in Custiodio et al. [5], as follows.

2.1 HP model

The hydrophobic–polar (HP) model is premised on the hydrophobic effect: during the folding process, hydrophobic amino acids will tend to group together in a hydrophobic core of the protein, hidden from the water solvent (which is polar). Conversely, the hydrophilic, polar amino acids will tend to lie on the surface of the protein exposed to water. The HP-model therefore abstracts the amino acid sequence to a binary sequence of monomers that are either hydrophobic (H) or hydrophilic/polar (P).

The HP lattice model for a specific protein is stored as a sequence of directions that are traversed in the 3D lattice to reach the next H or P monomer [5]. There are six possible directions from any lattice point: F, B, L, R, U, D - forward, back, left, right, up and down (Figure 1).

2.2 Fitness function

The energy of a conformation in the HP lattice is calculated as the negative of the number of hydrophobic–hydrophobic contacts, defined as two non-consecutive (i.e. non-bonded) H monomers occupying adjacent sites on the lattice. For the genetic algorithm, energy is equivalent to fitness: a lower energy implies a more optimal model. Optimization aims to find the best conformation for a given HP sequence (i.e. with the most H-H contacts) by modifying the direction sequence using the genetic operators, discussed in Section 2.5.

2.3 Initial population

From the specification of the sequence of H or P monomers for the protein chain under investigation, an initial population of molecular conformations is randomly generated and checked with a repair mechanism (discussed below). The initial population was set to 500 individuals and function evaluations limited to four million (equivalent to roughly two hundred thousand generations in this scheme). In cases of comparisons with other algorithms and where optimal energies are known, this upper bound was rarely reached.

2.4 Parental selection and replacement

Each generation creates ten new individuals, which may or may not be added to the population.

Parent conformation structures are chosen from the population with tournament selection [14, 15]. Four candidates for the tournament are selected randomly from the population; the tournament then proceeds by selecting one or two individuals (based on the operator being used) with the probability, P_i, of selecting the ith individual with fitness f_i from a population of size N given by

$$P_i = \frac{f_i}{\sum_{n=1}^{N} f_n} (2)$$

This strategy allows for the fittest individuals to be selected more often, while still preserving diversity in the population.

When new individuals are created from the population, members must be replaced to maintain a fixed population size. In the "crowding" strategy implemented, a distance metric is used to determine the closest matching individual in the population to the new member; if the new member has a lower energy than the closest member, the new member replaces it in the population. If they have equal values, then there is a fifty percent chance of replacement. In the final case the new individual is discarded. The distance metric is the distance metric error (DME) given by

$$DME = \sqrt{\sum_{i,j}^{N} \frac{p_i - q_j}{N(N-1)}} (3)$$

where summation is over the length of both the molecules, the value p_i is the magnitude of the ith coordinate of H monomer site in the first chain and q_j denotes the magnitude of the jth coordinate of the H monomer site on the second chain. This function is computationally expensive to evaluate: even using matrix operations, this step is still $O(n^2)$ and thus one of the biggest bottlenecks in the algorithm.

2.5 Genetic operators

There are six genetic operators that can be applied to a chain in the GAHP algorithm: two-point crossover (2X), multi-point crossover (MPX), local move (LM), segment mutation (SMUT), exhaustive search mutation (EMUT) and loop move (LPM). The crossover 2X and MPX operators are standard in genetic algorithms [15]. 2X conducts a two-point crossover of parents to produce two new members of the population. The MPX operator chooses multiple crossover points from the parent sequences (a random number of points between 2 and 10) and produces two offspring using alternating segments from each parent.

The LM, LPM, SMUT and EMUT operators are more complex, specific to the HP-problem, and are discussed in detail in Custodio et al. [5].

The LM operator swaps the directions of two consecutive monomers at a random location in the HP sequence. For example, if the moves for two consecutive monomers are *U,R* (up, then right), LM will alter the sequence to *R,U*.

The LPM operator works in a similar fashion, except that the monomers with swapped directions are not necessarily adjacent, but are chosen from two random locations on the monomer chain.

The SMUT operator works on a segment of molecule (of a random size between 2 and 7 monomers) and changes the direction of each monomer randomly to one of the six directions. Note that the random choice may be the same as the original value.

The EMUT operator selects a random monomer in the HP sequence and changes its direction to the best possible direction among the six according to the fitness function evaluation for each option. Only five evaluations are required, as the current conformation evaluation is already calculated in the previous step.

After application of any of the operators LM, LPM, SMUT and EMUT to a molecular HP chain, collision detection must be performed to ensure that the molecule is still in a valid conformation.

2.5.1 Dynamic application of operators

Operators are applied according to a dynamic probability [5] which is adjusted every generation, as follows. Whenever an operator creates an individual with better fitness than the current best in the population, that operator is rewarded with a numerical reward equal to the difference in HH contacts between the old best and current individual. The operators that created the parent of the individual are rewarded with half that amount (or a quarter to each if two parents were used in crossover rather than a single parent in mutation). This simple reward addition for the ith operator is given by

$$R_i = R_i + \left(f_{new_{best}} - f_{old_{best}}\right) \quad (4)$$

The probability is then

$$P(i) = \frac{R_i}{\sum_{n=1}^{6} R_n} \quad (5).$$

After the creation of ten new individuals, the probabilities of operators are adjusted to new values calculated as their current overall reward as fraction of the total reward for all operators. No probability can fall below five percent so as to eliminate it from use, a simple check subtracts the shortfall from the current highest probability to keep probabilities above five percent. The rewards for each operator are initialized to 1 to ensure uniformity. Finally, if an operator has not produced an optimal individual in five hundred calls to the method, then a penalty of one unit is given as a negative reward, while maintaining that the operator's reward stays above one. This mechanism ensures that operators that stagnate the population after some time are penalized and then allows for other operators to be more likely, eventually returning to a uniform distribution if no improvements are made. This has the benefit of exploring the fitness landscape more efficiently for a global minimum.

2.6 Collision detection and repair

Collision detection entails checking that set of directions for a given chain of monomers does not result in two monomers occupying the same lattice point. The collision detection algorithm [5] starts from a lattice coordinate $\{x,y,z\}$. Then, to calculate the next coordinate, a unit is added or subtracted from the relevant coordinate of the point (x or y or z) according to the direction listed (for example, a move of *F* adds 1 to x, whereas *B* subtracts 1). Each point calculated is added to a list of points. If a newly calculated point is already in the list, then a collision has occurred. If the molecule is traversed without any collisions, then it is a valid conformation.

A repair mechanism is used in the generation of new molecules for the initial population. In this procedure, a candidate is checked at each stage/monomer for a collision. If one occurs, the monomer is assigned a new random direction. This procedure repeats until either there is no longer a collision, or all directions lead to collisions. In the latter case, rather than removing the member from the population, the fitness of the individual is assigned to zero.

2.7 Validation and performance

To compare our AGAHP to the results reported for GAHP, we used the same data sets reported in original work, as follows. We used two sets of randomly generated sequences comprising ten sequences of protein molecules 48 monomers in length (numbered 48.1 to 48.10) [10] and ten sequences of proteins 64 monomers in length (numbered 64.1 to 64.10) [7]. In addition, we used five biologically inspired sequences comprising 46, 58, 103, 124 and 136 length monomers [12, 13]. In analyzing function evaluations, chains of 27 monomers were used [7]. All data sets are included in the open-source implementation.

In addition, we tested the algorithm further with randomly generated chains. First, we generated a data set of three protein chains 200 monomers in length: chain 200.1 has 100 hydrophobic monomers, 200.2 has 50, and 200.3 has 30. For each test case involving a specific molecule, a set of fifty runs (twenty for molecules of length 200) was recorded and the best result, the average of the best results and the standard deviation is quoted for the AGAHP.

Then we generated chains for benchmarking of the code. For this, chains of sizes 25, 50, 75, 100, 125... 200, 225, and 250 were generated, with 50% of monomers hydrophobic (H). The algorithm was executed with each chain on ten trials/runs with 3000 generations each and the time taken to execute the entire calculation is taken. This gives a good indication of the scalability of the algorithm with increasing chain length. Of course, runtime will vary according to the machine it is executed on, but this serves as an indication of the scalability of the algorithm with increasing monomer length.

Averages and standard deviations for fitness values of optimal structures predicted by the algorithm are calculated over 50 runs for the sequences of length 64 and 27 shown in Tables 3 and 4,

and over 20 runs for the sequences of length 200, with a population size of 500 and 3000 generations in each case. Repeated runs are necessary for exploring solution space, as the algorithm has many random components and varying initial conditions.

The performance of the algorithm is analyzed in a similar way to F.L. Custodio et al. [5]. A call to evaluate the fitness of a candidate in the population counts as a function call/evaluation. The number of function evaluations to reach an optimal (or best solution in the case no optimal is found) fitness for a given monomer sequence compared against the performance analysis conducted on GAHP (which have also conducted their own comparisons against other methods).

3. RESULTS AND DISCUSSION

Our reimplementation of the GAHP algorithm, AGAHP, produced comparable results to GAHP for the overwhelming majority of test cases (Table 1). For the set of 48- and 64-monomer chains (Table 1) AGAHP produced results that agree with GAHP on eight of the ten cases for the 48 length chains and 6 out of the 10 on the 64 length chains. Note that the results for GAHP are not necessarily optimal and that alternative algorithms or, indeed, additional experiments may identify more optimal structures. Molecular conformational optimization suffers to a considerable degree form the multiple-minima problem: as discussed in Custodio et al., the low standard deviation for the best fitness values over the fifty runs is indicative of the algorithm's tendency to become stuck in a tightly packed fitness landscape of local optima around a low-energy conformation. This is quite evident from the standard deviation associated with the mean of each fitness measurement: each experiment, with 50 trials/ runs, had a substantial deviation with respect to the quoted mean, indicating that there were widely varying best fitness values at the end of each of the trials. For example, using sequence 64.1, fitness values ranged from 21 to the optimal 31 and an average of 26.13. Because of the random nature of this algorithm, repeated experiments may produce different results. It is thus quite conceivable that, over many more trials, the algorithm may converge to a more optimal structure. Indeed, our implementation identifies structures with more H-H contacts than the original GAHP in some of the "biologically inspired" test cases: 46.1, 58.1 and 103.1. Further, in most cases the averages between AGAHP and GAHP do fall within the bounds of each other's standard deviations. However, on some of the test cases, (e.g. 48.3 and 64.6) the AGAHP averages trailed the GAHP minimum by two or three H-H contacts with a low standard deviation, indicative of a local minima traps.

Sample conformations for structures 64.5, 64.8 and 103.1 are shown in Figure 2. Note that a similar di-core structure of the 103.1 chain was also seen by Custodio et al, but our more optimal structure has two additional contacts.

We also calculated optimal structures for three randomly generated 200-monomer protein chains (Table 2 and Figure 3). The genetic algorithm was not previously tested on structures of this size. The 200.3 chain has the fewest hydrophobic monomers (30) and hence the most extended conformation (Figure 3c), while 200.1 with 50% hydrophobic monomers is the most compressed conformation, with a large hydrophobic core (Figure 3a).

Table 1: Comparison of best and average structures in terms of number of HH contacts reported for the original GAHP [3] and our AGAHP implementation. Standard deviations are shown in brackets and best estimates across both implementations are in bold.

No. monomers	*ID*	GAHP		AGAHP	
		Best	**μ (σ)**	**Best**	**μ (σ)**
48	*1*	**32**	30.72 (0.67)	**32**	29.98 (0.89)
	2	**34**	31.26 (0.59)	**34**	31.11 (0.78)
	3	**34**	32.08 (0.80)	32	30.41 (0.52)
	4	**33**	31.16 (0.81)	**33**	30.93 (0.94)
	5	**32**	30.52 (0.73)	31	29.81 (0.65)
	6	**32**	29.86 (0.78)	**32**	29.32 (0.86)
	7	**32**	29.82 (0.56)	**32**	28.32 (1.03)
	8	**31**	29.32 (0.58)	**31**	28.26 (1.15)
	9	**34**	31.92 (0.66)	**34**	30.98 (0.85)
	10	**33**	31.08 (0.56)	**33**	30.06 (0.68)
64	*1*	**31**	28.50 (1.10)	*30*	26.13 (0.95)
	2	**36**	33.18 (1.22)	**36**	32.25 (1.36)
	3	**44**	41.88 (0.87)	*43*	40.69 (0.56)
	4	**39**	36.02 (1.39)	**39**	35.80 (1.85)
	5	**40**	37.96 (1.12)	*38*	34.88 (1.10)
	6	**33**	31.52 (0.86)	*31*	28.55 (0.86)
	7	**28**	26.70 (0.70)	**28**	24.69 (1.03)
	8	**36**	33.72 (0.85)	**36**	32.26 (1.35)
	9	**38**	36.32 (0.93)	**38**	35.12 (1.25)
	10	**31**	28.90 (0.88)	**31**	28.51 (0.68)
46	*1*	35	33.04 (32.84)	**36**	33.23 (2.45)
58	*1*	42	40.04 (39.43)	**43**	38.56 (2.36)
103	*1*	50	46.58 (46.58)	**52**	47.67 (3.56)
124	*1*	**63**	58.12 (58.12)	**63**	56.55 (4.85)
136	*1*	**70**	62.22 (62.22)	*68*	60.32 (5.01)

Table 2: Best and average structures and number of function evaluations for AGAHP for three randomly generated structures of 200 monomers: 200.1 has 100 hydrophobic monomers, 200.2 has 50, and 200.3 has 30.

No. monomers	*ID*	Best	μ (σ)	No. f. eval.
200	*1*	98	87.15 (7.33)	465 300
	2	48	38.03 (6.46)	392 600
	3	29	21.68 (6.12)	367 100

The histograms of unique conformations with a specific number of H-H contacts for each of the three 200-monomer chains are graphed in Figure 4. As seen for shorter chains in the original work by Custodio et al, in each case there is a peak near mean number of H-H contacts for each chain, indicating that the algorithm identifies many suboptimal solutions very close to the optimal solution. An interesting additional feature we see for these larger chains is the skewness of the peaks near the mean. These indicate that, for larger chains, the algorithm finds substantially more members of lower fitness below the mean value, as it builds up to optimum and sub-optimum solution: the algorithm takes longer to explore this much larger fitness landscape. A further point is that, as the number of hydrophobic residues decrease from 200.1 to 200.3, the peak is less distinct and the number suboptimal conformations increases.

In terms of performance, the GAHP and AGAHP implementations show a similar number of fitness function evaluations (Table 3). Differences in the number of evaluations is again a largely statistical phenomenon due to the random nature of operator applicability, which results in different numbers of function evaluations depending on which type of operator (mutation or crossover) is called.

Table 3: Number of function evaluations reported for the original GAHP [3] and calculated for our AGAHP implementation for protein chains 64 amino acids in length.

No. monomers	*ID*	GAHP	AGAHP
64	*1*	228 000	198 600
	2	115 000	131 200
	3	87 000	94 400
	4	159 000	145 400
	5	134 500	149 500
	6	177 000	191 400
	7	76 500	71 600
	8	178 500	165 400
	9	74 500	103 200
	10	82 500	98 300

It is interesting to compare the genetic algorithm's performance on short 27-monomer chains with that reported for the particle swarm optimization (PSO) [7] and contact interactions (CI) [8] (Table 3), an analysis not done by Custodio et al., who found better performance for the longer-chains they used for comparison. Despite finding optimum conformations for all chains, the genetic algorithm does not always perform as well as competing models on short chains in terms of function evaluations. All though all algorithms found the same global minimum structures, the genetic algorithm in some cases requires many more function evaluations to achieve the same results. This poor performance on smaller length monomer chains might be attributed to the fact that the algorithm uses six operators together with dynamically varying probabilities, which might be an overly complex solution to a level of problem able to be solved by simpler methods. For example, the EMUT requires four fitness evaluations but may be an unnecessary operator for small problems. Therefore, the genetic algorithm performs poorly on smaller data sets, but out-performs other methods as the data size increases [5].

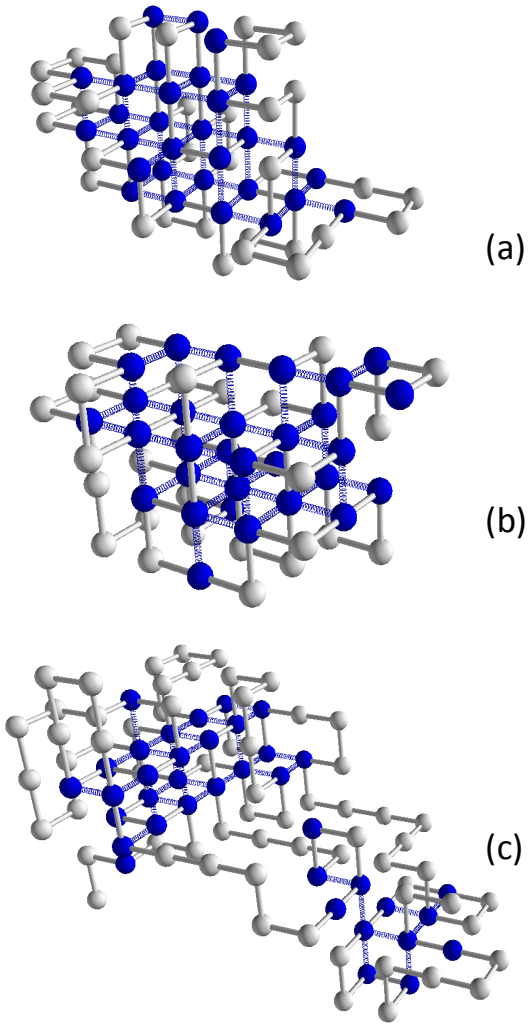

Figure 2: Best structures for selected sequences, with H-H contacts shown in blue. (a) 64.*5* (38 H-H contacts), (b) 64.*8* (36 contacts) and (c) 103.*1* (52 contacts).

Further, our analysis of the number of function evaluations is performed for longer chains of 200 monomers shows that the number of function evaluations can vary widely for different chains (Table 2). The number of evaluations tends to increase with increasing numbers of hydrophobic monomers for the three 200-monomer chains 200.1 has 100 hydrophobic monomers (465 300 function evaluations), and 200.3 has 30 (367 100 function evaluations).

Table 4: Number of function evaluations reported for the particle swarm optimization (PSO) [7] and contact interactions (CI) [8] methods and calculated for our AGAHP implementation for protein chains 27 amino acids in length.

No. monomers	*ID*	AGAPC	CI	PSO
27	*1*	6 870	15 854	3 158
	2	7 980	19 965	5 771
	3	9 410	7 991	2 667
	4	10 200	23 525	8 556
	5	4 920	3 561	893
	6	9 870	14 733	12 790
	7	13 560	23 112	17 024
	8	1 540	889	149
	9	2 270	5 418	1 915
	10	3 150	5 592	2 638

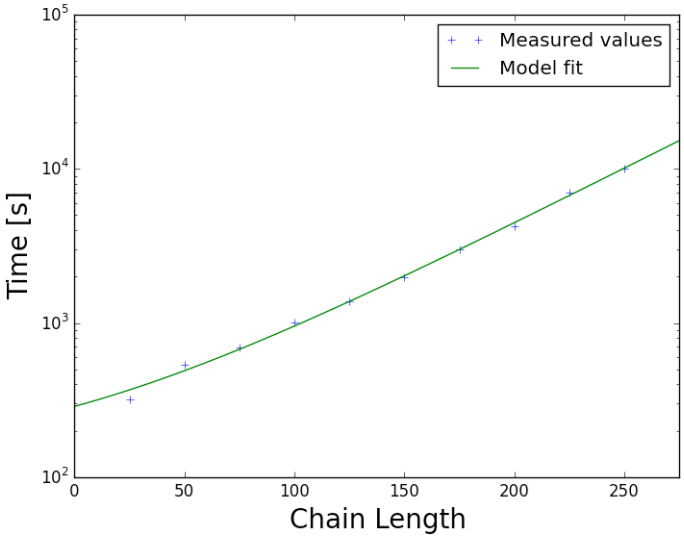

Figure 5: Plot of the real-time execution versus the length of the molecule chain, on logarithmic scale, for monomers ranging from length 25 to 250, together with fitted exponential model

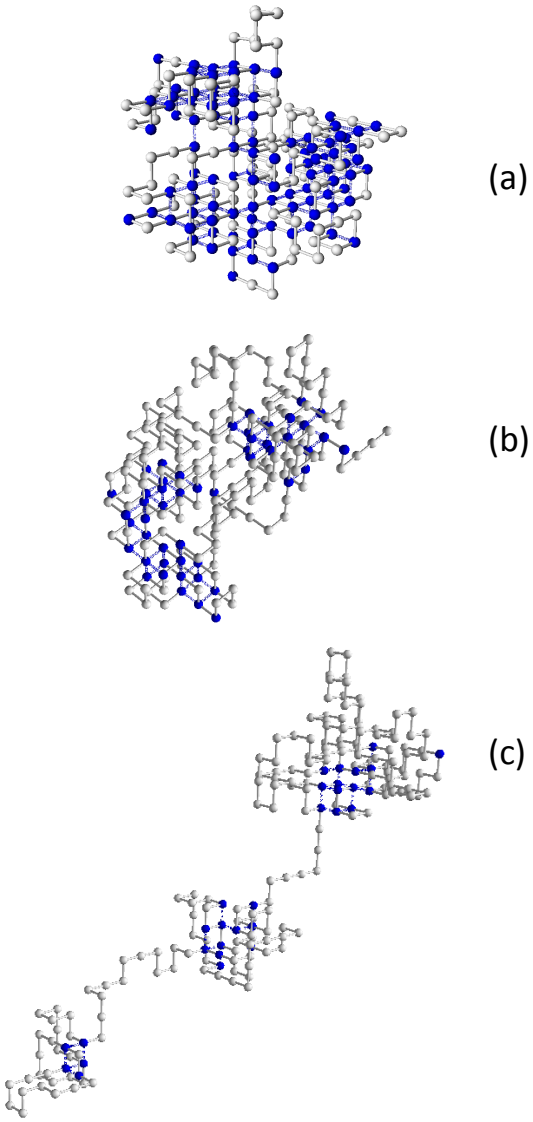

Figure 3: Optimal structures for chains (a) 200.1, (b) 200.2 and (c) 200.3. H-H contacts are shown in blue.

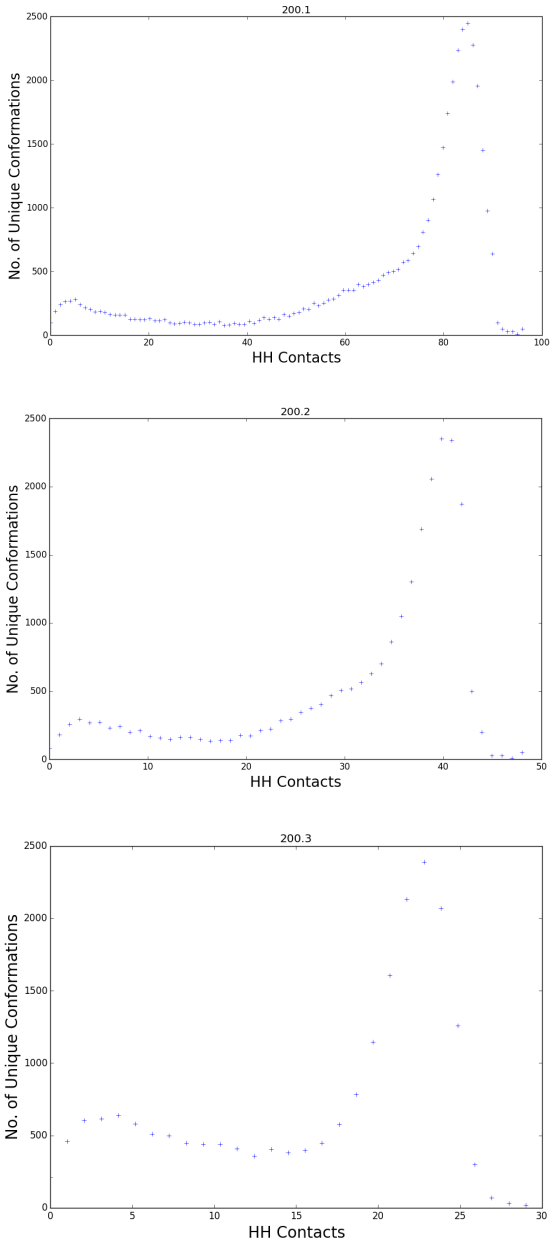

Figure 4: Plots of the number of unique conformations that appear over a 20 trial execution with 3000 generations using a pool size of 500 chains of length 200. The results are shown for the three 200-monomer chains; 200.1, 200.2 and 200.3

While the number of fitness function evaluations is the crucial measure of performance when considering the time it takes to reach global minimum [5,7,8], it is also useful to compare real-time analysis of performance, as the fitness function is not the only costly operation in the algorithm: the DME function is also expensive. The time required for execution on chains ranging from 25-250 monomers is plotted on a logarithmic scale in Figure 5. This shows that the real-time execution of the algorithm is exponential with respect to the length of the molecule chain: one can see an increase in the execution time data points which appears linear in this scaling regime. A Chi-square test of the data using an exponential model yields a good fit to the data. This analysis indicates that the genetic algorithm could benefit from a parallel genetic algorithm, [15] to reduce the time required to optimize long protein chains.

4. CONCLUSIONS AND FUTURE WORK

We have created an alternative and publicly available implementation of the GAHP genetic algorithm for protein structure prediction using the hydrophobic-hydrophilic lattice model. Our alternative implementation is freely available to the public at https://github.com/BrutishGuy/ADAGP.

We have tested this implementation on the data sets used for the original implementation, as well as on additional data sets and found that the results agree with those produced by GAHP on a large majority of the test cases. On the additional data sets, some novel structures were observed, such as di-core and tri-core hydrophobic cores.

Our analysis of the number of function evaluations shows equivalent performance to that reported for GAHP. The analysis on shorter length chains shows that the algorithm performs worse compared to other methods by requiring more function evalutations.

In addition, our real-time analysis of the execution on chains ranging in size from 40 -400 monomers shows a non-linear exponential increase in the real-time execution as the length of the chain increases, holding all other algorithm parameters constant. This points to promising future work of introducing a parallel genetic algorithm that may be run on multi-processor or multi-core architectures to achieve a speed-up for larger and more interesting chain structures and reduce the execution time needed.

In addition, extension of our implementation to a more sophisticated all-atom model, as in Custodio et al., will also be explored.

5. ACKNOWLEDGMENTS

The South African Medical Research Council funding agency provided financial support for this work.

6. REFERENCES

[1] Tramontano, A. 2005. Problem 4: Protein Structure Prediction. In *The Ten Most Wanted Solutions in Protein*

[2] *Bioinformatics, Chapman & Hall/CRC mathematical biology,* Etheridge, A., Gross, L., Lenhart, S., Maini, P., Safer, H. Voit, E., Eds.; CRC Press: Boca Raton, FL, 117-139.

[3] Chandru, V., DattaSharma, A. and Kumar, V.S.A. 2003. The algorithmics of folding proteins on lattices, Discrete Appl. Math. 127,1,145–161.

[4] Berger, B. and Leighton, T. 1995. Protein folding in the hydrophobic hydrophilic (HP) model is NP-complete. *Journal of Computational Biology,* 5, 27-40.

[5] Custodio, F. L., Barbosa, H. J. C., Dardenne, L. E. 2014. A multiple minima genetic algorithm for protein structure prediction, *Applied Soft Computing,* 15, 88-99.

[6] Bazzoli, A. and Tettamanzi, A. G. B.. 2004. Memetic algorithm for protein structure prediction in a 3D-lattice HP model, *Applications of Evolutionary Computing, EvoWorkshops,* 1-10.

[7] Mansour, N., Kanj, F. and Hassan, K., 2012. Particle Swarm Optimization Approach for the Protein Structure Prediction

in the 3D HP Model. *Interdisciplinary Sciences Computer Life Sciences,* 4, 190-200.

[8] Toma, L. and Toma, S. 1996. Contact interactions method: A new algorithm for protein folding simulations. *Protein Science*, 147-153.

[9] Cutello, V., Nicosia, G., Pavone, M., and Timmis, J. 2007. An immune algorithm for protein structure prediction on lattice models, *Trans. Evol. Comp.* 11, 1, 101–117.

[10] Shmygelska, A. and Hoos, H.H. 2005. An ant colony optimisation algorithm for the 2D and3D hydrophobic polar protein folding problem, *BMC Bioinformatics*. Vol. 6, 30.

[11] Hart, W., Krasnogor, N. , Smith, J. and Pelta, D. 1999. Protein structure prediction with evolutionary algorithms. *Proceedings of the Genetic and Evolutionary Computation Conference,* Vol. 23, 1596-1601.

[12] K. A. Dill, H. S. Chan and K. M. Fiebig, "Cooperativity in protien-folding kinetics," Proceedings of the National Academy of Sciences, USA, vol. 90, pp. 1942-1946, 1993.

[13] K. Yue, K. M. Fiebig, P. D. Thomas, H. S. Chan, E. I. Shaknhovich and K. A. Dill, "A test of lattice protein folding algorithms," Proceedings of the National Academy of Sciences, USA, pp. 325-329, 1995.

[14] Melanie, M. Introduction to Genetic Algorithms, Cambirdge, Massachusetts: The MIT Press, 1999.

[15] Davis, L. Handbook of Genetic Algorithms, Boston: London International Thomson Computer Press, 1